Modified METEOR Metric using N-gram Language, Wordnet Synonyms and Stemming

Bryce Golamco
School of Computing Science, Simon Fraser University
Burnaby, BC, Canada V5A1S6
bgolamco@sfu.ca

Abstract

This paper shows the implementation of the baseline of the METEOR metric system created by Lavie and Argarwal. It extends the METEOR using N-grams, Wordnet synonyms and Stemming to improve on the accuracy of the metric. The result of these additions to the METEOR metric did indeed improve the accuracy and can be shown through single sentences tests.

1 Introduction

If two output sentences from two different machine translation programs are translated from a single sentence from French to English, it can be easy to tell which translation is better even without any knowledge or understanding of the French sentence. For output sentences that need a decision of which output is better than the other ranging from thousands to a million sentences, using humans to decide on these would be a rather expensive and a long process. Automatic MT evaluations were proposed to mediate this problem. The most common evaluation metric used is BLEU, but for single sentences the BLEU metric is quite unusable due to its lack of recall using the brevity penalty as a substitute. The METEOR metric however is much more reliable for single sentences as it not only uses recall, it penalizes sentences with different word alignments from the reference translation. This paper focuses on extending the METEOR metric and combining different methods that are inspired from BLEU.

2 Approach

The approach used in this paper was by modifying the METEOR metric with an N-gram Language model, Wornet synonyms and Stemming to enchance the accuracy of the metric.

2.1 Baseline METEOR

The baseline METEOR metric decides on two translations by matching unigrams to an alignment with the target sentence and the reference sentence. The Meteor metric is calculated using the harmonic mean of unigram precision and recall. Computing the unigram precision requires the number of words in the target translation that can also be found in the reference translation. It is then divided by the number of unigrams in the target translation. This is shown in the formula below.

Precision:

$$P = \frac{c}{w}$$

C is the number of unigrams that matches.

W is the number of unigrams in the target.

The computation of recall requires the same as above however c is instead divided by the total number of unigrams in the reference translation as shown below as v.

Recall:

$$R = \frac{c}{v}$$

The Harmonic mean of the METEOR metric is then computed as the combination of both the unigram precision and recall with the formula below. The alpha is a parameter that is tunable and different in every language.

Harmonic Mean:

$$F_{mean} = \frac{PR}{\alpha \cdot P + (1 - \alpha) \cdot R}$$

The METEOR metric also penalizes the sentence with differing word order by looking at the set of unigrams. It counts the number of chunks which can be defined as the number of sets of unigrams in the target sentence with the same word ordering as in the reference sentence. It is then divided by the number of matched unigrams of the target to the reference.

$$Fragment = \frac{chunks}{c}$$

The penalty is then calculated as below with gamma and beta to be the parameters that differ with every language.

$$Penalty = \gamma \cdot Fragment^{\beta}$$

The parameters α, β and γ are tuned using hill climbing to be α = 0.8, β = 0.5, γ = 0.3 from testing.

The score of the METEOR metric can then be taken as:

$$score = (1 - Penalty) \cdot F_{mean}$$

When the score is compared to another score of a translated sentence, the higher score is better.

2.2 Extending the METEOR metric

The METEOR metric is then extended to improve upon the baseline.

2.2.1 N-gram Harmonic Mean

To extend the METEOR metric, the metric was then modified with an N-gram language model like that of the BLEU metric. Both precision and recall were given a N-gram model to incentivize longer matches of N-grams. To compute the N-gram precision, the sentence is first divided into their corresponding N-grams such that the number of N-gram matches from the target to the reference is divided by the number of N-grams in the candidate length. The formula is summarized as below:

$$Precision_n = \frac{c_n}{w_n}$$

$Precision_n$ is the N-gram precision

c_n is the number of word matches

w_n is the number of N-grams in the reference sentence

w_n can be expressed as the form:

$$w_n = number\_of\_words - n + 1$$

n would be the N-gram number (i.e. a tri-gram would be n = 3)

Example 1 - Computing the number of word matches of a Tri-gram

Target sentence: The brown fox jumps

Reference sentence: The quick brown fox jumps

The example above shows how to count the number of tri-grams in the target sentence that matches the reference. Now if we divide up the target sentence into tri-grams, we would get:

$$(The, brown, fox)(brown, fox, jumps)$$

Now the tri-gram for the reference sentence would be:

$$(The, quick, brown)(quick, brown, fox)(brown, fox, jumps)$$

In this example, the number of word matches that was found would be 1 since the tri-gram *(brown, fox, jumps)* was present in both the target and the reference sentences.

The number of word matches found would then be divided by the number of tri-grams in the target sentence which is w_n in the equation of the N-gram precision.

Calculating the N-gram Recall of the sentence is similar to the calculation of the N-gram model with the only difference is the division of the number of N-grams in the *reference* sentence rather than the target sentence.

2.2.2 N-gram Penalty Calculation

Since both precision and recall has been changed into an N-gram precision, we must also change the penalty involved in calculating the score of the sentence. Calculating the N-gram number of chunks of the target sentence requires the counts of sets of N-grams that match the reference sentence with the target sentence rather than unigrams. Going back to example 1, the number of chunks that would be counted would be 1 since there were only 1 set of

tri-grams which matches and has the same word ordering as in the reference sentence. A better example would be:

Example 2 - Tri-gram Chunks:

Target sentence: The brown fox jumps over in the lazy dog

Reference sentence: The quick brown fox jumps over the lazy dog

The number of chunks that would be found here would be 2. This is because the tri-gram sets that match the reference translation would be:

First chunk:

{(brown, fox, jumps), (fox, jumps, over)}

Second chunk:

{(the, lazy, dog)}

To calculate the penalty we simply plug it in the penalty equation with chunks as 2 and c as the number of N-gram word matches as described in Section 2.2.1.

2.2.3 Calculating the Score

The calculation for the Harmonic Mean of the N-gram is to simply use the formula as the baseline METEOR Harmonic Mean for each N-gram of the precision and recall. To calculate the total score of the metric would be to sum up all the respective Harmonic Mean multiplied by their penalty N-gram. The formula is as follows:

$$score = \sum_{n=1}^{4}(1 - Penalty_n) \cdot F_{mean_n}$$

2.2.4 Stemming and Wordnet

To improve the baseline even further, the Lancaster Stemming algorithm was used to stem the words of the sentences essentially replacing the words of both the target sentence and the reference with its stem counterparts. This way the words with the same stem would be treated as equal when the word matches are being counted.

The NLTK wordnet was also used to match the words that are synonyms in the target sentence with the reference sentence. The words in the target sentence are all checked if there is a word in the reference sentence which are synonyms and if it is so, it sets the two words as equal. Essentially, if a word in the target sentence is in the same synset as a word in the reference sentence, the two words are considered equal. The Stemming and Wordnet are both combined together with the N-gram model of the baseline metric to produce the current results given now.

3 Data

The data files used in this paper was the given default dataset which where the train-test.hyp1-hyp2-ref and hyp1-hyp2-ref data. These two datasets were used mainly for optimizing the parameters of the meteor to their single sentence variant parameters. The datasets were also used to compare the results between the different metrics created.

3.1 Tuning the parameters

Although the paper on the METEOR metrics by Lavie and Agarwal have found that the best parameters for the METEOR metric were those of which with:

α = 0.9
β = 3.0
γ = 0.5

The METEOR that Lavie and Agarwal tested on were generally long length sentences and unlike these single sentence tests. These parameters were tuned again using hill climbing to find the maximum output. The parameters from the hill climbing were found to be:

α = 0.8
β = 0.5
γ = 0.3

4 Code

The only code used to aid this project were from the Natural Language Toolkit Module of Python. The Lancaster Stemmer and the Wordnet was the result of using this Toolkit. The baseline Meteor, N-gram Language Model, Hill Climbing and Bleu was produced by myself.

5 Experimental Setup and Results

To show the comparisons between the different versions of METEOR metrics, each metric was tested with the two datasets which were hyp1-hyp2-ref and train-test.hyp1-hyp2-ref. The metrics that were tested were the baseline METEOR, Bleu, Meteor

with the N-gram language model, Meteor with N-gram language model combined with wordnet and Stemming. These tests were all single sentences and so the accuracy that is to be expected would be around 40 - 50% depending on the dataset. Once all the code has been run, they are compared with each other to determine the best evaluation method. The outputs for those metrics are shown in the table below.

Type of Metric	Accuracy
BLEU	22.68%
Baseline METEOR	51.73%
METEOR (N-gram only)	51.24%
METEOR (N-gram, Wordnet, Stem)	**52.04**%

Table 1: Test on hyp1-hyp2-ref data.

Type of Metric	Accuracy
BLEU	20.41%
Baseline METEOR	34.81%
METEOR (N-gram only)	35.39%
METEOR (N-gram, Wordnet, Stem)	**36.29**%

Table 2: Test on train-test.hyp1-hyp2-ref data.

There are many difference in these metrics; the BLEU metric uses the N-gram precision coupled with the brevity penalty instead of using recall. This makes longer matches more favourable. The implementation of BLEU here has all their N-grams weighted equally. The baseline METEOR metric was explained above and the difference to that of the METEOR with a N-gram Language Model is the N-gram where N is equal to 4. The Last one is the METEOR with a N-gram Language Model added with a Wordnet and Stemmer. This is possible by first preprocessing the strings of sentences with the Wordnet and Stemmer before running it with the METEOR N-gram.

6 Analysis of Results

It can be seen from the results in the table that the METEOR N-gram with Wordnet and Stemmer did improve over the baseline METEOR. However, it did not seem to be a huge improvement over the baseline. This could perhaps be attributed to the fact that the tests were run on single sentences rather than paragraphs of the texts.

7 Future Work

Fixes and additions that could have improved the project here was to tune the parameters using a hill climb search on the METEOR with the N-gram Language Model coupled with the Wordnet and Stemmer. Due to the CPU limitations and poor optimizations in the code, running the code with the dataset would take about 1 hour per iteration. Performing a hill climb test would usually take about 4000 iterations, and so was not feasible in a span of two weeks.

Adding more parameters to the N-gram precision and recall would also be beneficial as different N-grams may require a different value in its parameter. If we could have tested different parameters such as if precision with $N > 1$ in N-grams have lesser penalties, then it could have incentivize longer matches of sentences rather than the higher N-grams becoming too penalized for being too fragmented.

Another addition that would have also deemed helpful would be to have the code tested on datasets with more than single sentences. It would be helpful to be able to thoroughly test and check how much the Modified METEOR metric would improve over the baseline METEOR.

8 Conclusion

From the experimental setup and results, adding the N-gram Language model with the Wordnet's Synonym Matching and Stemming the strings to the baseline METEOR metric would indeed improve the accuracy of the metric. It can also be shown that the METEOR with only the N-gram Language model can also increase the baseline in some cases.

9 References

Banerjee, S., & Lavie, A. (2005, June). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments [PDF]. Pittsburgh: Carnegie Mellon University.

Lavie, A., & Argarwal, A. Eteor: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments [PDF]. Pittsburgh.

Papinero, K., Roukos, S., Ward, T., & Zhu, W. BLEU: a Method for Automatic Evaluation of Machine Translation [PDF]. New York: IBM T. J. Watson Research Center.

University, P. (2015, March 17). What is WordNet? Retrieved December 09, 2017, from http://wordnet.princeton.edu/